ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

(Laboratorio di)
Amministrazione di sistemi

Creare e gestire VM con Vagrant e SSH

Marco Prandini

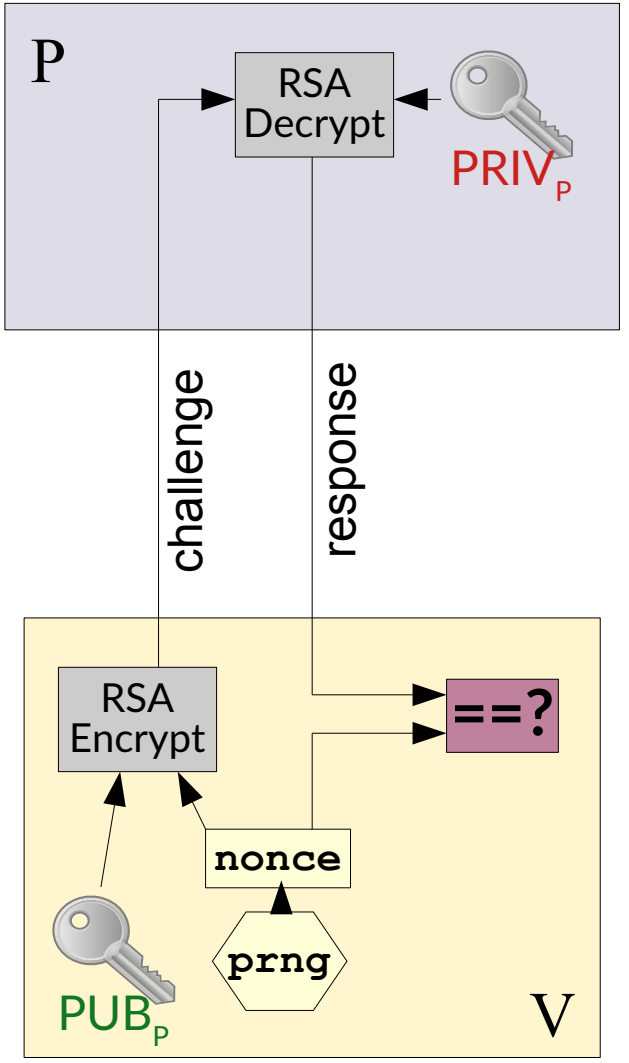
Dipartimento di Informatica – Scienza e Ingegneria

Secure Shell

- Necessità: amministrazione remota
- Predecessori: TELNET
  - Nessuna confidenzialità del canale
  - Nessuna autenticazione dell'host
  - Autenticazione passiva dell'utente

L'autenticazione a chiave pubblica

- Autenticazione - attori
  - PROVER (P) = colui che deve provare la propria identità
  - VERIFIER (V) = colui che deve accertare l'identità di P
- Autenticazione attiva a chiave pubblica
  - Operazione basata sulla *crittografia asimmetrica*
  - P dispone della propria *chiave privata* $PRIV_P$
    - è un elemento di identificazione perché non è condivisa con nessuno
  - V dispone della *chiave pubblica* di P PUB_P
    - è matematicamente legata a $PRIV_P$ ma non permette di ricavarla (!)
  - V può *sfidare* P a dimostrare di essere in possesso di $PRIV_P$
  - P conoscendo $PRIV_P$ può rispondere correttamente alla sfida
  - V può verificare la correttezza della risposta usando PUB_P



Secure Shell

- Il collegamento SSH tra client (ssh) e server (sshd) avviene attraverso questi passi essenziali
  - Negoziazione dei cifrari disponibili
  - Autenticazione dell'host remoto per mezzo della sua chiave pubblica
  - Inizializzazione di un canale di comunicazione cifrato
  - Negoziazione dei metodi disponibili per l'autenticazione dell'utente
  - Autenticazione dell'utente
- Ognuno dei passi elencati può essere portato a termine in modo configurabile, al fine di garantire il compromesso tra sicurezza e flessibilità più adatto al contesto.

Secure Shell – host authentication

- L'autenticazione dell'host remoto è importante per evitare di cadere nella trappola tesa da un eventuale uomo nel mezzo, che potrebbe così catturare la password dell'amministratore spacciandosi per l'host su cui egli vuole effettuare il login
  - Non è previsto un sistema centralizzato di attestazione dell'autenticità della chiave dell'host
  - Alla prima connessione l'amministratore deve utilizzare un metodo out-of-band per determinare la correttezza della chiave pubblica presentata dall'host
  - Alle connessioni successive la chiave pubblica memorizzata dal client dell'amministratore permette di effettuare un'autenticazione attiva

- Le chiavi pubbliche vengono memorizzate nel file known_hosts nella directory .ssh posta nella home dell'utente sul client.

Secure Shell – user authentication

- Ci sono due possibilità per l'autenticazione dell'utente sull'host remoto
  - Autenticazione passiva, tradizionale, con username e password – i dati sono trasmessi all'host autenticato su di un canale cifrato, quindi con buon livello di sicurezza
  - Autenticazione attiva, per mezzo di un protocollo challenge-response a chiave pubblica – presuppone che l'utente si doti della coppia di chiavi, e che installi correttamente sull'host remoto la chiave pubblica

Secure Shell – user authentication

- In entrambi i casi, l'identità dell'utente con cui viene tentato il login sull'host remoto può essere selezionata
  - in assenza di indicazioni specifiche verrà usato lo stesso nome utente con cui l'operatore sta lavorando sul client

  Es:
  - utente "marco" sul client esegue "ssh remoteserver"

    → si presenta come utente "marco" su "remoteserver" e si deve autenticare di conseguenza
  - utente "marco" sul client esegue "ssh root@remoteserver"

    → si presenta come utente "root" su "remoteserver" e si deve autenticare di conseguenza

Secure Shell – key generation

- Per poter effettuare l'autenticazione attiva un utente deve
  - generare una coppia di chiavi asimmetriche
    - comando **`ssh-keygen -t rsa -b 2048`**
  - installare sull'host remoto la chiave pubblica.
    - file locale **`.ssh/id_rsa.pub`**
    - copia su host remoto

      `scp .ssh/id_rsa.pub user@remote:`
    - append alla lista di utenti autorizzati (su *remote*)

      `cat id_rsa.pub >> .ssh/authorized_keys`

Secure Shell – avvertenze

Il ruolo autenticante della password viene sostituito dalla presenza della chiave privata dell'utente sul client – la segretezza della password è quindi sostituita dalla riservatezza del file che contiene la chiave privata

- Grande cura nell'impostazione dei permessi di file e directory (nota di tipo pratico: spesso il passwordless login non funziona semplicemente perché i permessi sulla directory *.ssh* dell'host remoto sono troppo larghi, e quindi il server sshd "non si fida" dell'integrità del suo contenuto)
- Possibilità di proteggere la chiave privata con una password

– Vi priva della possibilità di passwordless login

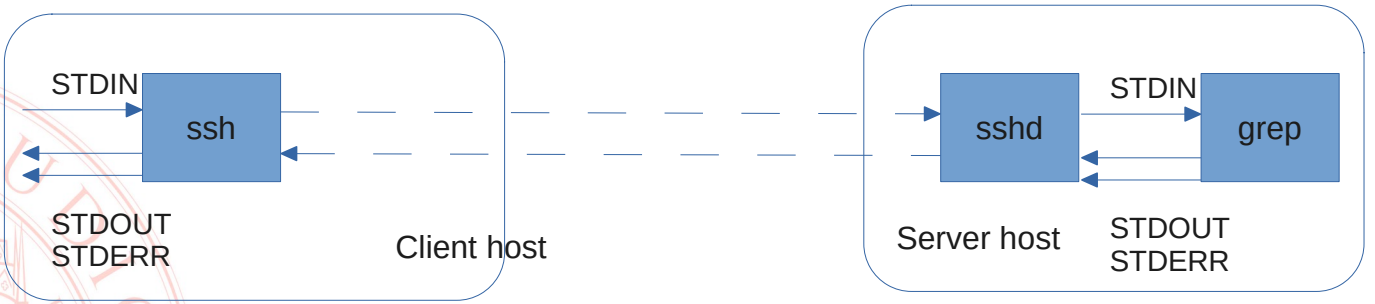
– Più sicuro comunque che utilizzare direttamente la password dell'account remoto, e più pratico se si amministrano molti host remoti

Secure Shell – esecuzione remota

- Lanciando *ssh utente@host* si ottiene un *terminale remoto* interattivo.
- Aggiungendo un ulteriore parametro, viene interpretato come **comando** da eseguire sull'host remoto al posto della shell interattiva; gli stream di I/O di tale comando vengono riportati attraverso il canale cifrato sul client.

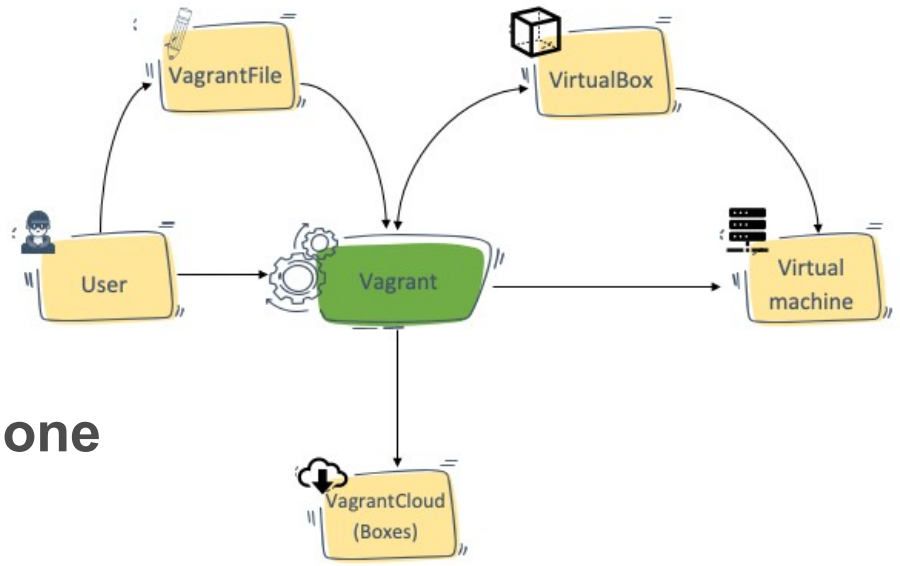
  Es: `ssh root@server "grep pattern"`

  - I dati forniti attraverso STDIN al processo ssh sul client vengono resi disponibili sullo STDIN del processo grep sul server
  - STDOUT e STDERR prodotti dal processo grep sul server "fuoriescono" dagli analoghi stream dal processo ssh sul client



Vagrant

- Finalità
  - **riproducibilità** affidabile di ambienti di test virtualizzati
  - **portabilità** rispetto allo specifico strumento di virtualizzazione
- Modello
  - l'immagine della VM di base (**box**) può essere facilmente depositata e successivamente reperita e importata da un deposito online (**VagrantCloud**)
  - I parametri per istanziare una VM sono definiti in un file di testo (**VagrantFile**)
  - Vagrant può utilizzare una varietà di sistemi di virtualizzazione (**Provider**) per eseguire la VM



Vagrantfile - un esempio

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

Vagrant.configure("2") do |config|

  config.vm.box = "ubuntu/trusty64"

  config.vm.network "private_network", ip: "192.168.33.10"
  config.vm.network "public_network"
  config.vm.network "forwarded_port", guest: 80, host: 8080

  config.vm.synced_folder "code/", "/app/code"

  config.vm.provider "virtualbox" do |vb|
    vb.memory = 2048
    vb.cpus = 1
  end

  config.vm.provision "shell", inline: <<-SHELL
    apt-get update
    apt-get install -y apache2
    service apache2 start
  SHELL

end
```

box di base

vari tipi di interfacce di rete virtuali

mappatura cartella host-guest

provider e configurazione hw virtuale

provisioner: azioni di configurazione da eseguire dentro la VM

Vagrant - operazioni di base

- Installare una box (opzionale)
  - i produttori di Vagrant mettono a disposizione della comunità un repository (HashiCorp's Vagrant Cloud box catalog), per le box presenti lì l'installazione è automatica al primo avvio
  - per scaricare manualmente la box in locale, in modo da non dover attendere all'avvio [o per specificare una fonte diversa dal default]:

    `vagrant box add autore/versione [ url ]`
  - documentazione altre operazioni sulle box

    https://www.vagrantup.com/docs/cli/box
- Inizializzare un Vagrantfile
  - tipicamente si crea una directory, e si lancia

    `vagrant init autore/versione`
  - viene creato un Vagrantfile coi default della box, modificabile
- Avviare / monitorare / fermare / distruggere una VM
  - dalla cartella che contiene il Vagrantfile, semplicemente

    `vagrant {up|status|halt|destroy}`

Vagrant + SSH

- `vagrant up`
  - alla prima esecuzione crea automaticamente una chiave privata sull'host, e installa la chiave pubblica nella VM
  - predispone una mappatura di rete da una porta host alla porta 22 guest
- `vagrant ssh`
  - utilizza automaticamente queste impostazioni per connettersi alla VM come utente `vagrant`, senza password
  - l'utente `vagrant` è abilitato a usare `sudo`
- `vagrant ssh-config`
  - mostra come configurare il client `ssh` standard per comportarsi come `vagrant ssh`, in modo da poterlo invocare in tutte le possibili varianti (incluso `scp`)